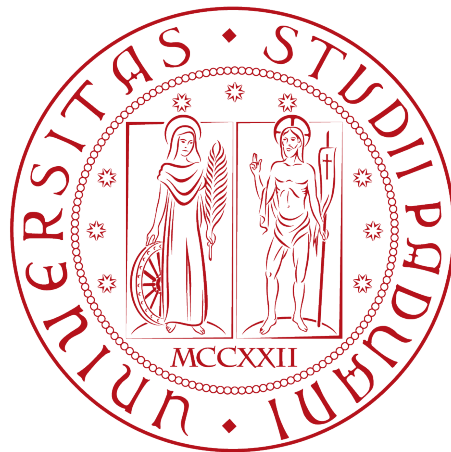


Università degli Studi di Padova

DIPARTIMENTO DI MATEMATICA "TULLIO LEVI-CIVITA"

CORSO DI LAUREA IN INFORMATICA



**Integrazione di Single Sign-On in Unix
Pluggable Authentication Module (Unix
PAM)**

Tesi di laurea

Relatore

Prof. Davide Bresolin

Co-relatore

Dr. Mattia Zago

Laureando

Ivan Antonino Arena

ANNO ACCADEMICO 2022-2023

“Quando desideri una cosa, tutto l’Universo trama affinché tu possa realizzarla.”

— Paulo Coelho

Dedicato alla mia famiglia

Abstract

Lo scopo della tesi è illustrare il lavoro eseguito con Athesys Srl, system integrator specializzato nello sviluppo di soluzioni di [Identity and Access Management \(IAM\)](#), in termini di innovazione e ricerca applicate all'ambito dell'integrazione di servizi web con sistemi UNIX-based. Nello specifico, si discute la collaborazione con il team infrastructure ed il team dedicato alla ricerca in materia di identità digitale e la conseguente realizzazione di un componente [Single Sign-On \(SSO\)](#) compatibile con Linux [Pluggable Authentication Modules \(PAM\)](#).

“No eternal reward will forgive us now for wasting the dawn.”

— Jim Morrison

Ringraziamenti

Innanzitutto, vorrei esprimere la mia gratitudine al Prof. Davide Bresolin, relatore della mia tesi, per l'aiuto e il sostegno fornitomi durante la stesura del lavoro.

In secondo luogo, vorrei ringraziare di cuore l'azienda ospitante, Athesys Srl, in particolare, il mio tutor esterno Mattia Zago, Roberto Griggio e Leonardo Speranzon, per la disponibilità e l'impegno con cui mi hanno affiancato durante il periodo di stage.

Desidero, inoltre, ringraziare con affetto i miei genitori per avermi appoggiato in ogni mia scelta durante il mio periodo universitario e per avermi fornito il supporto ed i mezzi per portarlo a termine con serenità.

Infine, ci terrei a ringraziare tutte le amicizie più significative che ho stretto a Padova, che mi hanno regalato gioie e sorrisi durante questi tre emozionanti anni.

Padova, Luglio 2023

Ivan Antonino Arena

Indice

1	Introduzione	1
1.1	L'azienda	1
1.2	Il progetto	1
1.3	Struttura della tesi	3
2	Studio tecnologico	4
2.1	Identity as a Service	4
2.1.1	Identity and Access Management	4
2.1.2	Single Sign-on	5
2.1.3	Linux Pluggable Authentication Modules	8
2.2	Self-Sovereign Identity	8
2.3	Tecnologie utilizzate	9
2.3.1	FreeIPA	9
2.3.2	Linux Containers	10
2.3.3	Secure Shell	10
2.3.4	Proxmox	11
3	Ricerca e sviluppo	12
3.1	Configurazione dello stato iniziale	12
3.2	Linux Pluggable Authentication Modules	14
3.2.1	Sviluppo del modulo PAM	14
3.3	FreeIPA Identity Provider	14
3.4	Configurazione Monokee	15
3.5	Configurazione FreeIPA	15
3.6	Problematiche riscontrate	16
3.7	Documentazione	16
3.8	Sviluppi futuri	17
4	Conclusioni	18
4.1	Raggiungimento degli obiettivi	18
4.2	Conoscenze acquisite	18
4.3	Valutazione personale	19
	Acronimi e abbreviazioni	20
	Bibliografia	22

Elenco delle figure

1.1	Logo di Athesys Srl	1
1.2	Logo di Monokee Srl	1
2.1	Diagramma di flusso dell'autenticazione con SAML	6
2.2	Diagramma di flusso dell'autorizzazione con OAuth2	7
2.3	Diagramma di flusso dell'autenticazione con OIDC	8
2.4	Logo di FreeIPA	10
3.1	Vista parziale del template download di LXC	13
3.2	Schermata di Proxmox con VM avviata	13
3.3	Schermata di creazione app OAuth2 da Monokee	15
3.4	Schermata di creazione OpenID provider da Monokee	15
3.5	Autenticazione con Monokee SSO tramite FreeIPA da CLI	16

Capitolo 1

Introduzione

In questo capitolo vengono descritti l'azienda ospitante ed il progetto dell'attività di stage curriculare.

1.1 L'azienda

Athesys Srl ([Figura 1.1](#)) è un'azienda di consulenza informatica nata a Padova nel 2010 "dalla sinergia di affermati professionisti del settore IT"[\[2\]](#), specializzata in ambito System Integration, Database Management, Sicurezza applicativa, Governance Cloud Platform, Hyperconvergenza e Sviluppo Software in modalità Agile.

Athesys comprende la spin-off Monokee[\[17\]](#) ([Figura 1.2](#)), fondata nel 2017 come soluzione cloud-based per la gestione dell'identità e dell'accesso ([Identity as a Service \(IDaaS\)](#)), la quale offre, come funzionalità principale, un sistema di [SSO](#) basato su diversi tipi di autenticazione, sia passwordless che tramite soluzioni di [Self-Sovereign Identity \(SSI\)](#).



Figura 1.1: Logo di Athesys Srl



Figura 1.2: Logo di Monokee Srl

1.2 Il progetto

Generalmente, per poter utilizzare un sistema, sia esso fisico, come, ad esempio, un computer, o virtuale, come un servizio web, è necessario disporre di un'identità digitale che certifichi che l'utente abbia il permesso di utilizzare tale sistema. Per associare la propria identità digitale al dispositivo o servizio che si vuole utilizzare occorre effettuare

un'autenticazione attraverso il sistema (o i sistemi) messi a disposizione dallo stesso. Il processo di autenticazione risulta, dunque, fondamentale e prioritario nel mondo dell'informatica. I sistemi basati su UNIX utilizzano un metodo di autenticazione relativamente semplice che sfrutta la tecnologia [Pluggable Authentication Modules \(PAM\)](#), memorizzando i dati degli utenti su dei file di testo locali. Esistono, però, anche metodi di autenticazione più complessi, come il [Single Sign-On \(SSO\)](#), che, per mezzo di diversi protocolli, consente ad un utente di autenticarsi a più servizi contemporaneamente utilizzando le stesse credenziali.

Il progetto dello stage nasce dall'esigenza di Athesys Srl di estendere le funzionalità rese disponibili da Monokee in ambito di gestione degli accessi e delle identità digitali. Monokee dispone di un [SSO](#) interno, che permette l'accesso canonico tramite nome utente e password, ma anche tramite credenziali biometriche, account Google e MetaMask. Queste funzionalità, tuttavia, sono disponibili unicamente attraverso l'interfaccia web, perciò il desiderio dell'azienda era quello di renderle disponibili anche dal terminale di macchine con sistemi operativi Linux-based, poiché da loro ampiamente utilizzate.

Il fine dell'attività di tirocinio era, dunque, quello di ricercare, ed, eventualmente, sviluppare, una soluzione che consentisse effettuare l'autenticazione al proprio account Monokee tramite il relativo [SSO](#) proprietario da riga di comando, in particolare, da macchine UNIX Debian e [Red Hat Enterprise Linux \(RHEL\)](#).

Il framework utilizzato per il progetto è FreeIPA, un gestore delle identità e degli accessi ([IAM](#)) gratuito ed open-source che combina varie tecnologie, quali Linux, [Lightweight Directory Access Protocol \(LDAP\)](#), [Massachusetts Institute of Technology \(MIT\) Kerberos](#), [Network Time Protocol \(NTP\)](#), [Domain Name System \(DNS\)](#) ed [System Security Services Daemon \(SSSD\)](#), e consta di un'interfaccia web e di strumenti di amministrazione tramite command-line[11].

Tramite FreeIPA è possibile configurare un server in cui vengono memorizzate tutte le informazioni relative agli utenti, come le credenziali, i ruoli ed i rispettivi privilegi, consentendo, così, di amministrare facilmente le identità digitali condivise attraverso una stessa rete di dispositivi, sostituendo un più ampio sistema di autenticazione a quello già presente nativamente sui dispositivi in questione (in questo caso [PAM](#) per i dispositivi Linux).

L'attività, dalla durata totale di circa trecento ore, si è sviluppata in, essenzialmente, tre fasi.

In una prima fase, mi sono occupato di configurare diverse macchine [Community Enterprise Operating System \(CentOS\)](#), [Red Hat Enterprise Linux \(RHEL\)](#) e Ubuntu, alcune delle quali messe a disposizione dall'azienda tramite la piattaforma open-source di virtualizzazione Proxmox, altre costruite localmente sul mio dispositivo con il modulo di virtualizzazione di Linux, [Linux Containers \(LXC\)](#), per poi installare e configurare FreeIPA su ognuna di esse, utilizzando una macchina come server e le altre come client, per verificare il corretto funzionamento della gestione delle identità e della comunicazione all'interno di una stessa rete.

Successivamente, mi sono dedicato alla ricerca di un metodo che consentisse di utilizzare il sistema di [SSO](#) di Monokee da terminale per autenticarsi. Inizialmente, ho studiato l'autenticazione tramite Linux [PAM](#) ed ho valutato la possibilità di sviluppare un modulo aggiuntivo per [PAM](#) che integrasse la funzionalità ricercata, aderendo ai protocolli di autenticazione maggiormente utilizzati. Tuttavia, ho deciso di non proseguire ulteriormente con questa opzione dopo aver trovato una soluzione più

efficiente, resa disponibile dall'applicativo di FreeIPA.

Nell'ultima fase del mio lavoro, infatti, ho sfruttato il sistema di autenticazione tramite Identity Provider esterno di FreeIPA, creando un utente associato al mio indirizzo e-mail di Monokee e configurando, lato Monokee, un'applicazione [Open Authorization 2.0 \(OAuth2\)](#) ed un provider [OpenID Connect \(OIDC\)](#), impiegando due protocolli (rispettivamente di autorizzazione e di autenticazione) ampiamente utilizzati in questo tipo di sistemi, per poi comunicare al server gli specifici indirizzi richiesti dai protocolli per verificare l'identità dell'utente da riga di comando.

Grazie questa soluzione, ogni utente di Monokee ha la possibilità di accedere al proprio account dal terminale di una qualunque delle macchine configurate con FreeIPA, utilizzando lo stesso [SSO](#) precedentemente disponibile solo attraverso l'interfaccia web, ed operare su di esse.

Infine, ho proseguito con lo sviluppo cercando di migliorare ulteriormente la soluzione trovata provando ad estendere la possibilità di accedere ai dispositivi già configurati da qualunque altro dispositivo, utilizzando la tecnologia [Secure Shell \(SSH\)](#). Tuttavia, non sono riuscito a terminare l'implementazione prima del raggiungimento del monte ore totale a disposizione.

In conclusione, sono riuscito ad implementare con successo una soluzione per poter utilizzare l'[SSO](#) di Monokee da riga di comando rispettando le richieste dell'azienda e lasciando loro anche una traccia documentativa per agevolare eventuali sviluppi futuri.

1.3 Struttura della tesi

I successivi capitoli della tesi sono strutturati come segue:

Il secondo capitolo illustra lo studio tecnologico effettuato spiegando nel dettaglio i concetti e gli strumenti impiegati;

Il terzo capitolo descrive i processi di ricerca attuati, lo sviluppo delle soluzioni trovate e la stesura della relativa documentazione;

Il quarto capitolo conclude, infine, la tesi riassumendo gli esiti del lavoro svolto e sviluppando le dovute conclusioni.

Capitolo 2

Studio tecnologico

In questo capitolo vengono illustrati nel dettaglio i concetti e le tecnologie utilizzate.

2.1 Identity as a Service

[IDaaS](#)[30], acronimo di Identity as a Service, è un modello di distribuzione dei servizi di gestione delle identità basato su cloud. Con [IDaaS](#), un'organizzazione può affidare la gestione delle identità dei propri utenti a un provider di servizi esterno, eliminando la necessità di mantenere un'infrastruttura locale per la registrazione degli utenti, l'autenticazione e l'autorizzazione degli utenti, la gestione delle password e la gestione delle sessioni.

I servizi [IDaaS](#) sono solitamente forniti come un'offerta di [Software as a Service \(SaaS\)](#), accessibili tramite internet e scalabili in base alle esigenze dell'organizzazione. I provider di servizi [IDaaS](#) gestiscono l'infrastruttura necessaria per garantire la sicurezza, la disponibilità e le prestazioni dei servizi di gestione delle identità.

Tra le funzionalità comuni offerte da questi servizi ci sono l'integrazione con directory aziendali esistenti, la possibilità di supportare l'autenticazione a fattori multipli, l'autenticazione federata per consentire l'accesso a risorse esterne all'organizzazione e la gestione centralizzata delle autorizzazioni degli utenti.

Utilizzando [IDaaS](#), le organizzazioni possono beneficiare di diversi vantaggi, tra cui una maggiore agilità nell'onboarding e nella gestione degli utenti, un'esperienza utente migliorata grazie a un'unica identità digitale per l'accesso a più servizi, una maggiore sicurezza grazie all'implementazione di misure di autenticazione avanzate e un risparmio sui costi e sulla complessità operativa associata alla gestione delle identità.

2.1.1 Identity and Access Management

[IAM](#)[29], è un insieme di processi, politiche, tecnologie e strumenti utilizzati per gestire l'identità digitale e il controllo degli accessi agli utenti all'interno di un'organizzazione. L'obiettivo principale di [IAM](#) è garantire che le persone giuste abbiano accesso alle risorse appropriate al momento opportuno, in modo sicuro e conforme alle politiche aziendali.

Alcune delle funzionalità chiave di [IAM](#) includono la gestione centralizzata delle identità, la gestione dei ruoli e delle politiche di accesso, la gestione delle password,

la gestione delle sessioni, la federazione dell'identità per consentire l'accesso a risorse esterne e l'integrazione con sistemi e applicazioni esistenti.

IAM consente anche di implementare il principio del "least privilege", che significa assegnare agli utenti solo i privilegi necessari per svolgere il proprio lavoro, minimizzando così il rischio di abusi o accessi non autorizzati.

L'implementazione di un sistema **IAM** efficace può portare a numerosi benefici per un'organizzazione, come una maggiore sicurezza dei dati, la conformità normativa, una migliore gestione delle risorse IT, una riduzione dei rischi e un'efficienza operativa migliorata.

2.1.2 Single Sign-on

L'**SSO**[32] è una tecnologia che consente agli utenti di accedere a più applicazioni e servizi utilizzando un'unica identità di accesso. In pratica, l'utente inserisce le proprie credenziali di accesso una sola volta e successivamente può accedere a tutte le applicazioni e servizi che supportano tale tecnologia senza dover inserire nuovamente le credenziali, alternativamente a come accade con l'autenticazione tradizionale. Ciò può risultare particolarmente vantaggioso se l'utente necessita di accedere a molte applicazioni o servizi diversi.

L'**SSO** funziona attraverso l'utilizzo di un'autorità di autenticazione centralizzata, chiamata **Identity Provider (IdP)**. L'**IdP** autentica l'utente e fornisce un token di sicurezza che contiene le informazioni sull'utente e sui servizi a cui ha accesso. Questo token può essere utilizzato per accedere a tutti i servizi che supportano tale sistema.

Per utilizzare l'**SSO**, le applicazioni e i servizi devono supportare uno dei protocolli **SSO** standard, come **SAML** o **OIDC**. Questi protocolli definiscono il modo in cui le informazioni di autenticazione dell'utente vengono trasmesse tra le diverse applicazioni e servizi.

L'**SSO** offre, dunque, numerosi vantaggi, tra cui una maggiore comodità per gli utenti, una maggiore sicurezza attraverso l'utilizzo di token di sicurezza a breve termine e una maggiore efficienza nella gestione delle identità e delle autorizzazioni degli utenti. Tuttavia, l'**SSO** richiede una pianificazione e una configurazione adeguata per garantire la sicurezza e la protezione dei dati degli utenti.

Security Assertion Markup Language

SAML[12], acronimo di Security Assertion Markup Language, è uno standard di autenticazione e autorizzazione basato su XML utilizzato per consentire la condivisione sicura di informazioni di autenticazione tra un fornitore di identità (Identity Provider) e un fornitore di servizi (Service Provider).

SAML è stato sviluppato per consentire l'integrazione tra applicazioni e servizi web di diverse organizzazioni, consentendo agli utenti di accedere a tali servizi con un'unica identità digitale senza dover creare account separati per ogni servizio.

Il protocollo **SAML** funziona utilizzando il concetto di token di sicurezza chiamato "asserzione". Un'asserzione **SAML** contiene informazioni sull'identità dell'utente autenticato, come il nome utente, l'ID utente, i ruoli e altre attribuzioni. Queste asserzioni sono scambiate tra il fornitore di identità e il fornitore di servizi utilizzando messaggi XML.

Il flusso di lavoro tipico di **SAML** coinvolge tre componenti principali: l'utente, il fornitore di identità e il fornitore di servizi. L'utente cerca di accedere a un servizio protetto dal fornitore di servizi. Il fornitore di servizi invia una richiesta di auten-

ticazione al fornitore di identità. Il fornitore di identità autentica l'utente e genera un'asserzione **SAML** contenente le informazioni sull'identità. Questa asserzione viene quindi inviata al fornitore di servizi, che verifica l'asserzione e consente l'accesso al servizio richiesto (Figura 2.1).

SAML è ampiamente utilizzato per consentire l'autenticazione e l'autorizzazione federate in scenari come il Single Sign-On (**SSO**), ma è anche utilizzato in scenari di federazione di identità, in cui più organizzazioni collaborano per consentire l'accesso sicuro a risorse condivise tra i partecipanti.

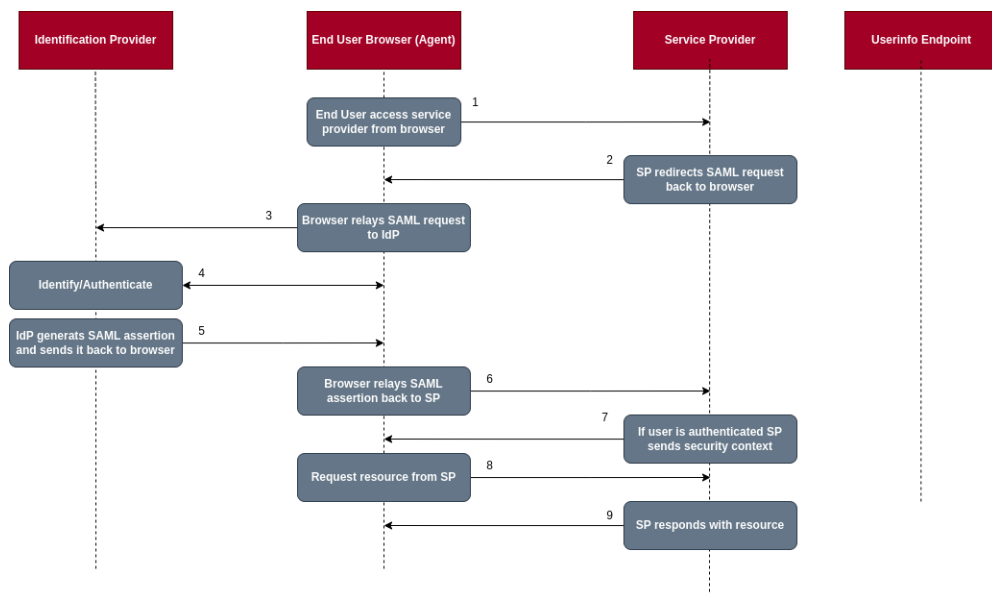


Figura 2.1: Diagramma di flusso dell'autenticazione con SAML

OAuth2

OAuth2[31] è un protocollo di autorizzazione che consente a un'applicazione di accedere alle risorse di un utente senza richiedere le credenziali dell'utente - e, di conseguenza, senza memorizzarle - nato per assicurare l'accesso sicuro e controllato ai dati di un utente da parte di applicazioni di terze parti.

Funziona attraverso una serie di flussi di autorizzazione, in cui l'utente concede l'autorizzazione all'applicazione per accedere alle sue risorse. L'applicazione, a sua volta, ottiene un token di accesso che può essere utilizzato per accedere alle risorse dell'utente (Figura 2.2).

Il protocollo **OAuth2** è utilizzato da molte grandi piattaforme online come Google, Facebook e Twitter ed è diventato, di fatto, uno standard nei servizi cloud, nei social network, nei servizi di pagamento online e in molti altri contesti.

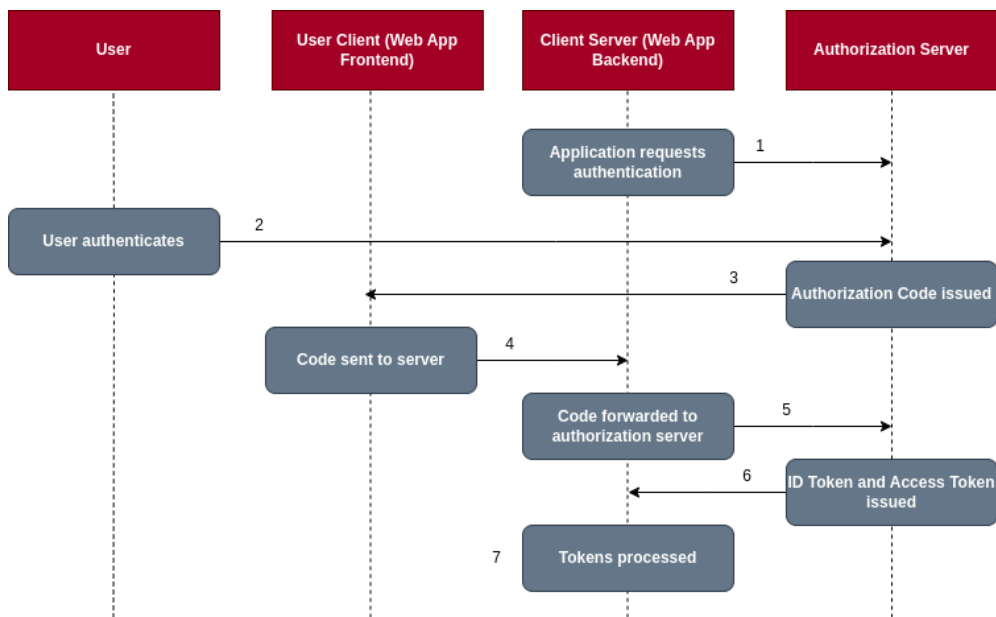


Figura 2.2: Diagramma di flusso dell'autorizzazione con OAuth2

OpenID Connect

[OIDC\[21\]](#) è un protocollo di autenticazione basato su [OAuth2](#), utilizzato per l'autenticazione degli utenti in applicazioni web e mobile. Progettato per risolvere il problema dell'autenticazione sicura e decentralizzata in applicazioni di terze parti, consente agli utenti di utilizzare l'[SSO](#) per accedere a diverse applicazioni, senza, quindi, dover creare un nuovo account per ogni applicazione, bensì delegando l'autenticazione ad un provider esterno (OpenID Provider).

[OIDC](#) fornisce un framework standard per l'autenticazione basata su JSON Web Tokens (JWT), in cui l'utente viene autenticato una sola volta e poi viene rilasciato un token di accesso contenente le informazioni di base dell'utente, come l'identificatore univoco, il nome e l'e-mail, che può essere utilizzato per accedere alle risorse protette ([Figura 2.3](#)).

Questo protocollo è stato adottato da molte grandi piattaforme online, tra cui Google, Microsoft e Amazon ed è anche supportato da molte librerie di sviluppo, caratteristica che lo rende semplice da implementare per gli sviluppatori.

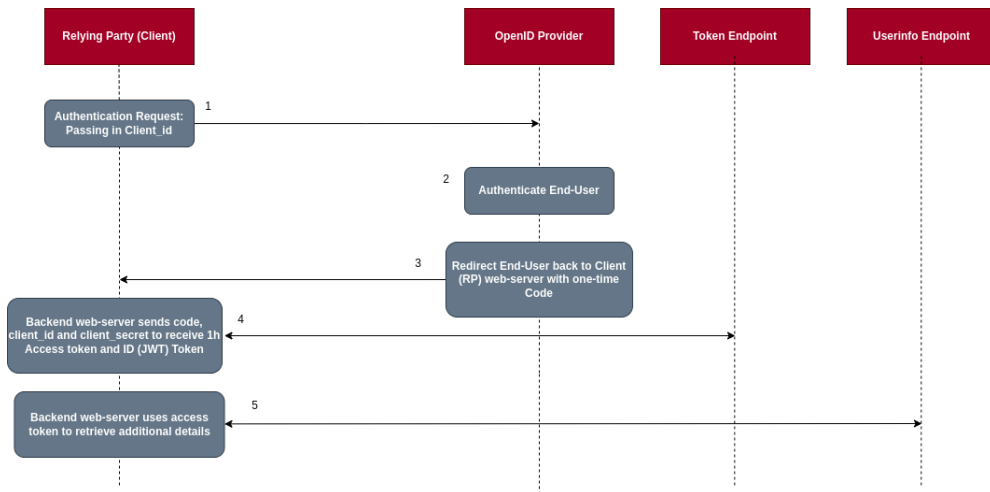


Figura 2.3: Diagramma di flusso dell'autenticazione con OIDC

2.1.3 Linux Pluggable Authentication Modules

Linux **PAM** (Pluggable Authentication Modules) è un framework di autenticazione per i sistemi operativi Linux e UNIX che consente di configurare diversi metodi di autenticazione, come quella tramite password, a due fattori, basata su token, biometrica, ecc.

Utilizzato in una vasta gamma di applicazioni e servizi, tra cui il sistema di login del sistema operativo ed il server **SSH**, il framework di **PAM** è composto da una serie di moduli, ognuno dei quali implementa una particolare funzionalità di autenticazione. I moduli **PAM** sono progettati per essere "pluggable", ovvero possono essere facilmente sostituiti o aggiunti senza dover modificare il codice sorgente del sistema operativo.

L'architettura modulare di **PAM** consente di creare una catena di moduli, in cui ciascuno dei quali può verificare una parte dell'identità dell'utente. Ad esempio, un modulo può verificare la password dell'utente, mentre un altro può verificare il certificato del client. Se uno qualsiasi dei moduli nella catena fallisce, l'intero processo di autenticazione viene interrotto. Inoltre, è possibile sviluppare dei moduli personalizzati ed integrarli nelle diverse funzioni che richiedono **PAM**.

2.2 Self-Sovereign Identity

La **Self-Sovereign Identity (SSI)**[24] è un nuovo approccio alla gestione delle identità digitali che consente agli utenti di possedere, controllare e condividere le proprie informazioni di identità in modo sicuro e privato. A differenza dei sistemi di identità tradizionali, in cui le informazioni di identità sono conservate in modo centralizzato da terze parti, l'**SSI** consente agli utenti di essere i proprietari esclusivi dei propri dati di identità digitali.

L'**SSI** si basa sulla tecnologia blockchain, che consente di creare registri distribuiti di informazioni sicure e immutabili. In tal modo, le informazioni di identità degli utenti vengono conservate in modo decentralizzato e sicuro, senza la necessità di un'autorità centralizzata di controllo.

Per utilizzare l'[SSI](#), gli utenti creano un'identità digitale che include le informazioni di identità necessarie, come nome, indirizzo e informazioni di contatto. Questa identità digitale viene conservata sulla blockchain e protetta da una chiave privata unica, che solo l'utente possiede.

Gli utenti possono utilizzare la propria identità digitale [SSI](#) per accedere a servizi online e condividere le proprie informazioni di identità solo con le parti che desiderano. Questo viene fatto attraverso l'utilizzo di un protocollo di scambio di informazioni sicuro e decentralizzato, chiamato [Decentralized Identifier \(DID\)](#).

L'[SSI](#) offre numerosi vantaggi, tra cui un maggiore controllo e privacy per gli utenti rispetto ai sistemi di identità tradizionali, una maggiore sicurezza attraverso l'utilizzo della tecnologia blockchain e una maggiore efficienza nella gestione delle identità digitali. Tuttavia, è ancora una tecnologia emergente e richiede una maggiore adozione e sviluppo per diventare un approccio mainstream alla gestione delle identità digitali.

2.3 Tecnologie utilizzate

Durante il periodo di stage ho avuto modo di utilizzare tecnologie per me nuove, oltre che approfondire la conoscenza di altre. Nelle sezioni seguenti sono illustrate tali tecnologie e strumenti.

2.3.1 FreeIPA

[FreeIPA](#)^[11] ([Figura 2.4](#)) è una soluzione open-source gratuita (GNU General Public License) di gestione dell'identità e dell'accesso per ambienti di rete basati su Linux/UNIX, originariamente sviluppato dalla comunità Fedora ed ora supportato da diverse organizzazioni, tra cui Red Hat e la FreeIPA Foundation. Consiste in un insieme di servizi integrati, che consentono di centralizzare l'autenticazione, l'autorizzazione e la gestione degli utenti e delle risorse in un'organizzazione.

FreeIPA è progettato per semplificare la gestione dell'identità e dell'accesso in ambienti di rete complessi, con molti utenti e computer. Consente agli amministratori di gestire facilmente l'accesso degli utenti a risorse e applicazioni, di delegare i privilegi di amministrazione e di definire ed applicare politiche di sicurezza coerenti in tutta la rete, come, ad esempio, limitare l'accesso alle risorse in base al ruolo dell'utente. Per fare ciò, mette a disposizione, oltre che agli strumenti della [CLI](#), un'interfaccia utente web intuitiva per la gestione degli utenti, dei gruppi e delle risorse della rete. Inoltre, FreeIPA è altamente scalabile e può essere distribuito su più server per gestire grandi reti.

Per l'autenticazione degli utenti, FreeIPA utilizza il protocollo Kerberos: gli utenti possono accedere alle risorse della rete utilizzando le loro credenziali Kerberos, senza dover inserire le password ogni volta. Per archiviare e gestire le informazioni sugli utenti, i gruppi e le risorse della rete, invece, utilizza il server di directory open-source 389 Directory Server, il quale offre funzionalità avanzate di ricerca, replica e sincronizzazione.

FreeIPA supporta l'autenticazione [SSO](#) tramite i protocolli [Security Assertion Markup Language \(SAML\)](#) (Security Assertion Markup Language) e [OIDC](#): ciò significa che gli utenti possono accedere a più applicazioni utilizzando le stesse credenziali di accesso.



Figura 2.4: Logo di FreeIPA

2.3.2 Linux Containers

LXC è l'acronimo di Linux Containers, un sistema di virtualizzazione basato sul kernel Linux che consente di eseguire più sistemi operativi isolati su una singola macchina host. A differenza della virtualizzazione completa, in cui ogni sistema operativo guest ha accesso all'intero hardware dell'host,

Utilizza la virtualizzazione basata sui contenitori, in cui ogni sistema operativo guest condivide le risorse hardware dell'host. La condivisione del kernel fa sì che i container siano molto leggeri e veloci e che abbiano un overhead di risorse molto basso rispetto ad altre tecnologie di virtualizzazione.

LXC fornisce un'interfaccia di riga di comando per la gestione dei container, che consente di creare, avviare, fermare, eliminare e gestire i container in modo semplice ed efficiente. Inoltre, supporta la creazione di immagini di container, che possono essere utilizzate per creare nuovi container in modo rapido e semplice.

Durante l'attività di stage ho utilizzato principalmente container basati su immagini **CentOS**, una distribuzione Linux basata su **RHEL** particolarmente adatta all'uso in ambiente server, che offre una vasta gamma di funzionalità e strumenti per gestire un'infrastruttura IT.

In particolare, ho utilizzato l'ultima versione di **CentOS Stream** (versione 9), che, a differenza della versione standard, riceve gli aggiornamenti in tempo reale durante lo sviluppo di **SSI**, consentendo agli utenti di testare e fornire feedback sulle nuove funzionalità e correzioni di bug in anteprima[5].

Un'altra differenza tra **CentOS** e **CentOS Stream** è la durata del supporto: **CentOS** ha, storicamente, fornito un supporto a lungo termine per le versioni rilasciate; la versione Stream, al contrario, è progettata per essere una piattaforma di sviluppo in continuo aggiornamento e non offre, dunque, il supporto a lungo termine.

2.3.3 Secure Shell

Secure Shell (**SSH**) è un protocollo di rete crittografato utilizzato per la gestione sicura di dispositivi di rete e per l'accesso remoto a sistemi informatici. Il protocollo **SSH** fornisce un canale di comunicazione sicuro tra due dispositivi, garantendo l'integrità, la riservatezza e l'autenticità delle informazioni trasmesse.

L'autenticazione avviene attraverso l'uso di chiavi pubbliche e private: in questo metodo, un'entità che desidera accedere a un sistema remoto genera una coppia di chiavi, una pubblica e una privata; la chiave pubblica viene fornita al sistema remoto, mentre la chiave privata viene conservata dall'entità; quando l'entità si connette al sistema remoto, la chiave privata viene utilizzata per autenticare l'entità.

Inoltre, **SSH** utilizza la crittografia per proteggere i dati trasferiti tra i dispositivi. In particolare, il protocollo utilizza la crittografia a chiave simmetrica per proteggere i dati durante la trasmissione, e la crittografia a chiave pubblica per autenticare le parti coinvolte.

2.3.4 Proxmox

Proxmox è una piattaforma di virtualizzazione open-source che combina la virtualizzazione basata su container (**LXC**) e la virtualizzazione basata su macchine virtuali (**Kernel-based Virtual Machine (KVM)**) in un'unica soluzione. È progettato per consentire la creazione, la gestione e l'esecuzione di macchine virtuali e container su un unico sistema, fornendo un ambiente flessibile e scalabile per le infrastrutture IT.

Una delle caratteristiche principali di Proxmox è la sua interfaccia di gestione Web, chiamata **Proxmox Virtual Environment (PVE)**, che fornisce un'interfaccia grafica intuitiva per la gestione delle risorse di virtualizzazione. **LXC** consente agli amministratori di creare, configurare e monitorare le macchine virtuali e i container, nonché di gestire l'archiviazione, la rete e altre risorse di sistema.

Proxmox offre, inoltre, funzionalità avanzate come la migrazione live delle macchine virtuali, che consente di spostare una macchina virtuale in esecuzione da un nodo all'altro senza interruzioni di servizio. Ciò offre flessibilità nella gestione delle risorse e consente di bilanciare il carico di lavoro tra i nodi di virtualizzazione.

La piattaforma è anche integrata con la gestione dell'archiviazione e della rete, consentendo la creazione e la gestione di storage e reti virtuali all'interno dell'ambiente di virtualizzazione: questo semplifica la gestione delle risorse e offre una maggiore flessibilità nella configurazione dell'infrastruttura di rete e storage.

Capitolo 3

Ricerca e sviluppo

In questo capitolo vengono descritti i processi di ricerca e sviluppo di una soluzione efficace per l'implementazione dell'[SSO](#) Monokee su Linux

3.1 Configurazione dello stato iniziale

Avendo già familiarità con Ubuntu ho, inizialmente, verificato che il pacchetto del server di FreeIPA fosse presente su questa distribuzione: ho presto appreso che esso era discontinuo sulle ultime versioni del sistema operativo.

Così, ho deciso di optare per [CentOS](#) Stream 9, ultima versione disponibile, per l'ottima compatibilità con FreeIPA e la migliore usabilità in ambienti server.

A questo punto, con l'aiuto del team di Athesys Srl, ho configurato [LXC](#) sulla mia macchina Ubuntu 22.04 [Long-term Support \(LTS\)](#).

Tramite il comando `lxc-create -t download -n ipa-server`, ho creato un nuovo container con il nome di `ipa-server` indicando di voler scaricare il template dalla lista di quelli disponibili, dalla quale ho scelto l'immagine di [CentOS](#) Stream 9 ([Figura 3.1](#)).

Dopo la creazione della macchina ho lanciato i comandi `lxc-start ipa-server` e `lxc-console ipa-server`, rispettivamente per avviare il container e per accedere al relativo terminale.

Dopo aver configurato il container per il server ed aver eseguito l'aggiornamento dei pacchetti con il comando `yum update`, sono passato all'installazione del server di FreeIPA, disponibile su quella release con il pacchetto `freeipa-server`.

Al mio arrivo negli uffici di Athesys Srl, l'azienda aveva già configurato per me un account sulla loro piattaforma di testing, `test.monokee.com`, utilizzando come e-mail il mio indirizzo istituzionale e garantendomi l'accesso a tutte le risorse della piattaforma, oltre che alla documentazione aziendale.

Inoltre, avevano predisposto sull'intranet aziendale, tramite Proxmox, delle macchine virtuali [CentOS](#) e [RHEL](#) pronte all'uso ([Figura 3.2](#)).

```
root@ubuntivan:~  
root@ubuntivan:~# lxc-create -t download -n ipa_server  
Downloading the image index  
---  
DIST  RELEASE ARCH  VARIANT BUILD  
---  
alpine 3.14 amd64 default 20230510_13:01  
alpine 3.14 amd64 default 20230510_13:05  
alpine 3.14 arm64 default 20230510_13:05  
alpine 3.14 armhf default 20230510_13:02  
alpine 3.14 i386 default 20230510_13:02  
alpine 3.14 ppc64el default 20230510_13:00  
alpine 3.14 s390x default 20230510_13:00  
alpine 3.15 amd64 default 20230511_15:54  
alpine 3.15 arm64 default 20230511_16:01  
alpine 3.15 armhf default 20230511_17:13  
alpine 3.15 i386 default 20230511_15:48  
alpine 3.15 ppc64el default 20230511_16:35  
alpine 3.15 s390x default 20230511_15:44  
alpine 3.16 amd64 default 20230511_15:57  
alpine 3.16 arm64 default 20230511_18:15  
alpine 3.16 armhf default 20230511_20:58  
alpine 3.16 i386 default 20230511_15:19  
alpine 3.16 ppc64el default 20230511_15:57  
alpine 3.16 s390x default 20230511_15:53  
alpine 3.17 amd64 default 20230511_22:07  
alpine 3.17 arm64 default 20230512_06:37  
alpine 3.17 armhf default 20230511_21:01  
alpine 3.17 i386 default 20230511_15:56  
alpine 3.17 ppc64el default 20230511_16:24  
alpine 3.17 s390x default 20230511_15:47  
alpine 3.18 amd64 default 20230511_15:45  
alpine 3.18 arm64 default 20230511_18:03  
alpine 3.18 armhf default 20230511_21:55  
alpine 3.18 i386 default 20230511_15:54  
alpine 3.18 ppc64el default 20230511_16:39
```

Figura 3.1: Vista parziale del template download di LXC

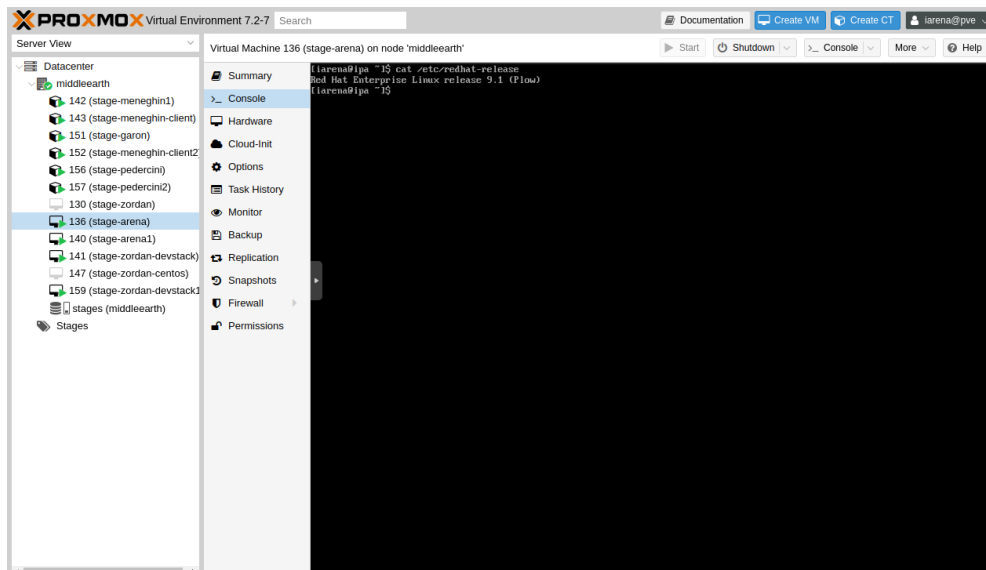


Figura 3.2: Schermata di Proxmox con VM avviata

3.2 Linux Pluggable Authentication Modules

La prima idea che ho avuto per integrare l'SSO di Monokee via SSH sul container CentOS che ho predisposto è stata quella di creare un nuovo modulo PAM. Ciò perché, studiando i file presenti al percorso `/etc/pam.d/`, ho trovato il file `sshd`, che stabilisce i moduli da utilizzare per autenticazione, autorizzazione, sessione e gestione della password. In un primo momento mi sono concentrato sulla parte di autenticazione, controllando il file di configurazione `common-auth`, incluso in `/etc/pam.d/sshd`, che rappresenta l'autenticazione predefinita di UNIX (con password memorizzata localmente).

Tuttavia, l'installazione di FreeIPA sovrascrive il parametro `UsePam yes` del file `/etc/ssh/sshd_config`, antepoendo dei parametri relativi a Kerberos, in modo da potersi autenticare con la password dell'utente FreeIPA specificato nel prefisso della macchina nel comando di SSH.

La mia idea era, dunque, quella di rimuovere questi parametri e tornare all'autenticazione via PAM, sostituendo però il modulo predefinito con uno creato appositamente per l'SSO di Monokee.

Il problema restava quello del riconoscimento dell'utente ma avevo già pensato a diversi modi in cui poterlo risolvere, così ho deciso di proseguire e sperimentare con lo sviluppo di un modulo PAM di test, per verificare la fattibilità della mia intuizione.

3.2.1 Sviluppo del modulo PAM

Dapprima, ho deciso di sviluppare una semplice applicazione PAM-aware[22], ovvero compatibile con Linux PAM, utilizzando il linguaggio C e facendo riferimento alla documentazione trovata[34][26][16][15].

Successivamente, ho sviluppato il modulo PAM di prova[23][20] e l'ho impostato come metodo di autenticazione per l'SSH con il seguente procedimento[33]: prima di tutto, ho modificato il file di configurazione `/etc/ssh/sshd_conf` disattivando il parametro `Set PasswordAuthentication` ed attivando il parametro `Set UsePam`; in seguito, ho modificato il file di configurazione dei moduli PAM da utilizzare per il servizio SSH, `/etc/pam.d/sshd`, commentando tutte le righe che facevano riferimento all'autenticazione ed inserendo una riga con il nome del modulo di prova che ho sviluppato, etichettato come `auth sufficient`, indicando che era sufficiente ottenere esito positivo da tale modulo per autenticarsi con successo.

3.3 FreeIPA Identity Provider

Dato che la soluzione con il modulo PAM si è rivelata essere più laboriosa del previsto ed avendo scoperto una funzionalità analoga nell'applicativo di FreeIPA, ho deciso di provare a configurare l'SSO con Monokee utilizzando quest'ultimo.

Navigando nell'interfaccia web del software, infatti, ho notato che nella sezione *Authentication > Identity Provider Servers* era possibile definire un IdP che utilizzasse OAuth2 2.0 come protocollo di autenticazione[10].

A questo punto, con l'aiuto del team, e, in particolare, del Chief Technology Officer (CTO) di Athesys Srl, mi sono spostato sull'infrastruttura di testing di Monokee per configurare un'applicazione OAuth2 da poter utilizzare come Identity Provider per FreeIPA.

3.4 Configurazione Monokee

All'interno dell'ambiente di test di Monokee, tramite l'interfaccia web, ho creato una nuova applicazione OAuth2[19] (Figura 3.3) ed un nuovo OIDC provider (Figura 3.4), che fornisce gli end-point per l'autenticazione via OIDC[18].

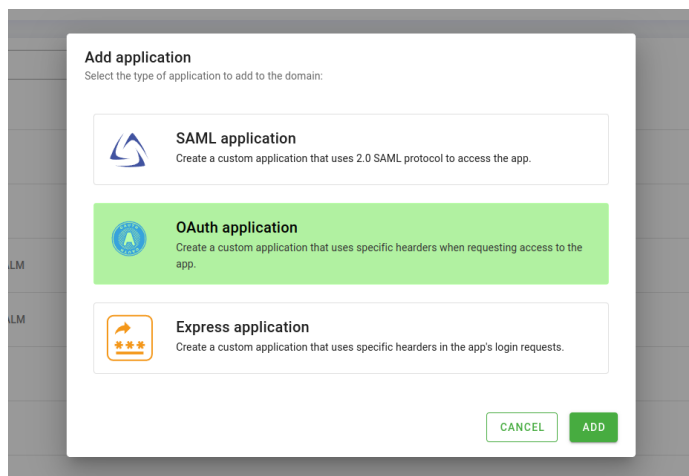


Figura 3.3: Schermata di creazione app OAuth2 da Monokee

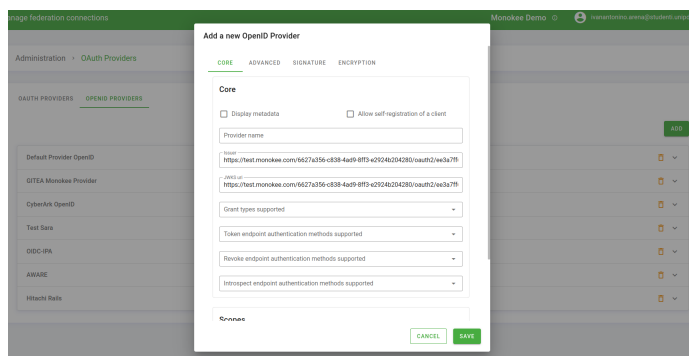


Figura 3.4: Schermata di creazione OpenID provider da Monokee

3.5 Configurazione FreeIPA

Da FreeIPA, ho proceduto a configurare un nuovo Identity Provider server, tramite la sezione della [Graphical User Interface \(GUI\)](#) di cui sopra, inserendo tutte i metadati richiesti, facendo riferimento all'applicazione OAuth2 creata su Monokee e agli end-point forniti dall'OpenID provider configurato precedentemente.

Successivamente, ho creato un utente FreeIPA che utilizzasse come unico metodo di autenticazione quella tramite Identity Provider esterno (*External IdP*), scegliendo Monokee come *IdP* e come identificatore il mio indirizzo e-mail istituzionale, già associato al mio account Monokee, per poter eseguire l'accesso via [SSO](#) con le mie credenziali[27].

Configurata correttamente l'infrastruttura di autenticazione sia su Monokee che su FreeIPA, ho proceduto a verificarne il funzionamento seguendo le indicazioni della documentazione relativa: dapprima, ho generato un file per l'autenticazione tramite canale FAST con il comando `kinit -n -c ./fast.ccache`; successivamente, ho richiesto l'autenticazione anonima tramite PKINIT con il comando `kinit -T ./fast.ccache monokee1`; a questo punto, viene eseguito il flusso di [OIDC](#) e viene mostrato un URL al quale autenticarsi tramite [SSO](#) di Monokee; ad autenticazione eseguita, basta tornare sul terminale e premere invio per completare l'autenticazione.

Per verificare la corretta autenticazione ho poi lanciato il comando `klist`, che mostra i ticket Kerberos richiesti, e controllato che l'utente attuale fosse lo stesso con cui volevo autenticarmi e che il ticket fosse valido ([Figura 3.5](#)).

```
[larena@ipa ~]$ kinit -n -c ./fast.ccache
[larena@ipa ~]$ kinit -T ./fast.ccache monokee1
Authenticate at https://test.monokee.com/6627a356-c838-4ad9-8ff3-e2924b284280/oauth2/7c7c28f2-411a-44ba-b5e0-a373bd75107e/device?user_code=Xyhw-HksZ and press ENTER.
[larena@ipa ~]$ klist
Ticket cache: FILE:/tmp/krb5cc_1000
Default principal: monokee1@ARENA.STAGE

Valid starting    Expires          Service principal
04/28/2023 14:33:54  04/29/2023 13:56:33  krbtgt/ARENA.STAGE@ARENA.STAGE
[larena@ipa ~]$
```

Figura 3.5: Autenticazione con Monokee SSO tramite FreeIPA da CLI

3.6 Problematiche riscontrate

Durante l'integrazione dell'[SSO](#) tramite FreeIPA, nonché già dal processo di configurazione dello stesso, ho riscontrato numerose problematiche.

In primis, a rendere il processo più faticoso del previsto, è stata la mancanza di una documentazione esaustiva e di informazioni utili in rete in merito alla risoluzione degli errori, dovuta probabilmente alla userbase di FreeIPA, che, benché tale servizio sia lo standard in ambito [IAM](#), è alquanto ridotta.

In secondo luogo, mi sono imbattuto errori di inconsistenza piuttosto limitanti tra i registri di FreeIPA e quelli di sistema, riscontrati anche da altri utenti in rete[3] [13] [7] [8] [9] [4], oltre che difetti di compatibilità della piattaforma con alcune versioni di [CentOS](#) e [RHEL](#).

Infine, l'autenticazione di FreeIPA tramite Monokee [SSO](#) non riesce a sovrascrivere quella di UNIX quando si vuole raggiungere una macchina tramite [SSH](#) su un utente Monokee già configurato sul server. Tuttavia, ho raggiunto il monte ore stabilito prima di poter arrivare ad una soluzione per questa problematica; l'attuale implementazione, dunque, comprende la possibilità di accedere al proprio ambiente Monokee tramite [SSO](#) da una certa macchina client unicamente se si è autenticati all'interno di essa con un utente locale di qualunque tipologia.

3.7 Documentazione

Athesys Srl ha richiesto la redazione di una guida che illustrasse il processo di configurazione del server FreeIPA e dei sistemi di Monokee per l'integrazione del [SSO](#) sulle macchine UNIX, per fornire una base documentativa per facilitare le future progressioni e sperimentazioni relative. Seguendo le indicazioni dell'azienda, ho utilizzato il formato Markdown, versionando il codice sul repository GitHub aziendale fornitomi dalla stessa[6]. La documentazione consta di un singolo file Markdown di 188 righe, completo di comandi, codice ed immagini esplicative.

3.8 Sviluppi futuri

A partire dai risultati che ho ottenuto durante l'attività di stage, gli sviluppi futuri possibili sono molteplici.

Innanzitutto, comincerei con il risolvere il problema legati all'accesso alle macchine con il server di FreeIPA installato tramite [SSH](#) su un utente di FreeIPA, e, dunque, per mezzo del sso di Monokee. L'[SSO](#), difatti, viene bypassato e viene richiesta la password dell'utente, la quale, di fatto, non esiste perché non è utilizzata da FreeIPA sugli utenti da autenticare con Monokee.

Dunque, andrei a verificare le impostazioni attive nei file di configurazione di [SSH](#), come `/etc/ssh/sshd_config`.

Risolto questo problema e avendo, quindi, una macchina a cui è possibile accedere tramite [SSH](#) direttamente con un utente Monokee, il prossimo passo potrebbe essere quello di predisporre delle macchine server con le risorse dell'azienda e fornirne l'accesso direttamente da Monokee, utilizzando un servizio come Apache Guacamole.

In questo modo, un utente Monokee privilegiato potrebbe accedere a delle macchine server gestire accessi, privilegi ed altro, oppure, nel caso di un utente subordinato, accedere a delle macchine client, gestite da quella server.

Capitolo 4

Conclusioni

In quest'ultimo capitolo vengono riportate le conclusioni e gli esiti dell'attività di stage

4.1 Raggiungimento degli obiettivi

L'attività è stata svolta quasi totalmente in linea con la pianificazione prevista: non ci sono stati ritardi di alcun tipo ed il primo periodo, quello di studio delle tecnologie, ha richiesto meno tempo di quanto preventivato, consentendomi, così, di approfondire ulteriormente lo sviluppo delle soluzioni trovate nelle fasi successive. Ho completato tutti gli obiettivi richiesti con successo, ad inclusione di quelli desiderabili e opzionali: dopo aver implementato con successo l'SSO di Monokee in una macchina CentOS utilizzando FreeIPA, ho prodotto la documentazione relativa, illustrando le procedure da seguire per replicare l'integrazione su altre macchine.

4.2 Conoscenze acquisite

Grazie allo stage con Athesys Srl mi sono addentrato in un campo dell'informatica che poco conoscevo: quello della sicurezza. Ho avuto modo di conoscere i concetti e le tecnologie più significative del momento presente in ambito di identità digitale, come l'SSI, l'SSO ed alcuni dei protocolli di autenticazione ed autorizzazione più diffusi, apprendendo, anzitutto, cosa significa creare e gestire un'identità digitale e quali sono i rischi di sicurezza legati ad essa. Oltre ad una già ampia formazione teorica, ho avuto anche la possibilità di migliorare le mie competenze in ambito di sistemi UNIX, entrando a contatto con parti di codice che cambiano direttamente il comportamento del sistema operativo, come i moduli PAM e l'SSH. Inoltre, ho imparato a creare dei container LXC dalle immagini messe a disposizione e, successivamente, a configurare un server di IAM, quale FreeIPA, modificando anche, in alcuni casi, manualmente dei file di sistema. Infine, ho messo in atto le conoscenze acquisite nella prima fase dell'attività, in particolare quelle riguardanti il funzionamento di OAuth2 ed OIDC, per implementare il Proof of Concept (PoC) richiesto.

4.3 Valutazione personale

Dopo circa trecento ore passate al fianco del team di Athesys Srl sono convinto di aver acquisito delle conoscenze e delle competenze, non tutte strettamente tecniche, fondamentali per il mio ingresso prossimo nell'industria: questa esperienza, che costituisce la mia prima nell'ambito del percorso che mi appartiene, quello dell'informatica, mi ha permesso di affrontare personalmente e toccare con mano le sfide, i problemi, le metodologie ed i traguardi propri della realtà delle aziende informatiche.

A partire dalla comunicazione, dall'organizzazione e dalla gestione del tempo e delle risorse, arrivando poi agli effettivi processi risolutivi e di sviluppo, sento di aver ricevuto un supporto significativo e di essermi messo alla prova, applicandomi al meglio in un ambiente a me quasi del tutto sconosciuto, al di fuori della mia zona di comfort, avendo, al contempo, contribuito produttivamente alle attività di Athesys Srl. Ora, al momento della stesura di questo documento, ripercorrendo ciò che ho fatto durante questa attività di stage, mi rendo conto ancora meglio del valore che essa ha avuto ed ha per me e per la mia carriera.

Acronimi e abbreviazioni

CentOS Community Enterprise Operating System . 2, 10, 12, 14, 16, 18, 20

CLI Command-Line Interface. 9, 20

CTO Chief Technology Officer. 14, 20

DID Decentralized Identifier. 9, 20

DNS Domain Name System. 2, 20

GUI Graphical User Interface. 15, 20

IAM Identity and Access Management. iv, 2, 4, 5, 16, 18, 20

IDaaS Identity as a Service. 1, 4, 20

IdP Identity Provider. 5, 14, 15, 20

KVM Kernel-based Virtual Machine. 11, 20

LDAP Lightweight Directory Access Protocol. 2, 20

LTS Long-term Support. 12, 20

LXC Linux Containers. 2, 10–12, 18, 20

MIT Massachusetts Institute of Technology. 2, 20

NTP Network Time Protocol. 2, 20

OAuth2 Open Authorization 2.0. 3, 6, 7, 14, 15, 18, 20

OIDC OpenID Connect. 3, 5, 7, 15, 16, 18, 20

PAM Pluggable Authentication Modules. iv, 2, 8, 14, 18, 20

PoC Proof of Concept. 18, 20

PVE Proxmox Virtual Environment. 11, 20

RHEL Red Hat Enterprise Linux . 2, 10, 12, 16, 20

SaaS Software as a Service . 4, 21

SAML Security Assertion Markup Language. 5, 6, 9, 21

SSH Secure Shell. 3, 8, 10, 14, 16–18, 21

SSI Self-Sovereign Identity. 1, 8–10, 18, 21

SSO Single Sign-On. iv, 1–3, 5–7, 9, 12, 14–18, 21

SSSD System Security Services Daemon. 2, 21

Bibliografia

Siti web consultati

- [1] *A REST interface for FreeIPA Plugged In*. URL: https://www.admin-magazine.com/Archive/2016/34/A-REST-interface-for-FreeIPA#article_12.
- [2] *Athesys Srl*. URL: <https://athesys.it/>.
- [3] *Bug 1602410*. URL: https://bugzilla.redhat.com/show_bug.cgi?id=1602410.
- [4] *CentOS Forums - Thread*. URL: <https://forums.centos.org/viewtopic.php?f=47&t=79922>.
- [5] *Comparing Centos Linux and CentOS Stream*. URL: <https://www.centos.org/cl-vs-cs/>.
- [6] *Documentazione stage*. URL: <https://github.com/monokee-stage/stage-unipd-arena/blob/main/docs.md>.
- [7] *FreeIPA users mailing list - Thread 1*. URL: <https://lists.fedoraproject.org/archives/list/freeipa-users@lists.fedorahosted.org/thread/GFGYCBUEFVU7XDIGHSAWF36MK6224ZVL/>.
- [8] *FreeIPA users mailing list - Thread 2*. URL: <https://lists.fedoraproject.org/archives/list/freeipa-users@lists.fedorahosted.org/thread/GZ5ISOSFY034DZILJFCPOGKBBGD5RBHI/>.
- [9] *FreeIPA users mailing list - Thread 3*. URL: <https://lists.fedoraproject.org/archives/list/freeipa-users@lists.fedorahosted.org/thread/L3F25NOAS5WZVYTGQAKSGVRWIEOB6EOG/>.
- [10] *FreeIPA's documentation*. URL: <https://freeipa.readthedocs.io/en/latest/index.html>.
- [11] *FreeIPA's website*. URL: https://www.freeipa.org/page/Main_Page.
- [12] *How SAML authentication works*. URL: <https://auth0.com/blog/how-saml-authentication-works/>.
- [13] *Issue 7473*. URL: <https://pagure.io/freeipa/issue/7473>.
- [15] *Linux man pages online*. URL: <https://man7.org/linux/man-pages/index.html>.

- [16] *Linux PAM configuration tutorial*. URL: <https://web.archive.org/web/20190420035810/https://fedetask.com/linx-pam-configuration-tutorial/>.
- [17] *Monokee Srl*. URL: <https://monokee.com/en/homepage/>.
- [18] *Monokee's documentation*. URL: <https://test.monokee.com/docs/>.
- [19] *OAuth 2.0 authorization flow*. URL: <https://www.loginradius.com/blog/engineering/authorization-code-flow-oauth/>.
- [20] *oidc.c*. URL: <https://github.com/monokee-stage/stage-unipd-arena/blob/main/pam/oidc.c>.
- [21] *OIDC's documentation*. URL: <https://openid.net/connect/>.
- [22] *pam_example.c*. URL: https://github.com/monokee-stage/stage-unipd-arena/blob/main/pam/pam_example.c.
- [23] *pam_module_example.c*. URL: https://github.com/monokee-stage/stage-unipd-arena/blob/main/pam/pam_module_example.c.
- [24] *Self-Sovereign Identity: The Ultimate Guide 2023*. URL: <https://www.dock.io/post/self-sovereign-identity>.
- [25] *Talking to FreeIPA JSON web API via curl*. URL: <https://adam.younglogic.com/2010/07/talking-to-freeipa-json-web-api-via-curl/>.
- [26] *Understanding PAM*. URL: <https://aplawrence.com/Basics/understandingpam.html>.
- [27] *Using external identity providers to authenticate to IdM*. URL: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html/configuring_and_managing_identity_management/assembly_using-external-identity-providers-to-authenticate-to-idm_configuring-and-managing-idm#proc_enabling-an-idm-user-to-authenticate-via-an-external-idp_assembly_using-external-identity-providers-to-authenticate-to-idm.
- [28] *Using OAuth 2.0 to Access Google APIs*. URL: <https://developers.google.com/identity/protocols/oauth2>.
- [29] *What is identity and access management? Guide to IAM*. URL: <https://www.techtarget.com/searchsecurity/definition/identity-access-management-IAM-system>.
- [30] *What is identity-as-a-service (IDaaS)?* URL: <https://www.cloudflare.com/en-gb/learning/access-management/what-is-identity-as-a-service/>.
- [31] *What is OAuth 2.0*. URL: <https://auth0.com/intro-to-iam/what-is-oauth-2>.
- [32] *What is Single Sign-On*. URL: <https://www.ibm.com/topics/single-sign-on>.
- [33] *Writing a Linux PAM module*. URL: <https://web.archive.org/web/20190523222819/https://fedetask.com/write-linux-pam-module/>.

- [34] *Writing a PAM-aware application*. URL: <https://web.archive.org/web/20190420073246/https://fedetask.com/writing-a-linux-pam-aware-application/>.